การประยุกต์ใช้เทอร์โบปาสคาล 4.0 กับ CAI ในวิชาฟิสิกส์

อาจหาญ สัตยารักษ์

วิทยาการคอมพิวเตอร์ที่เจริญขึ้นทำให้เราต้องติดตามเทคโนโลยีใหม่ตลอดเวลา การทำกราฟิกบนเครื่องคอมพิวเตอร์ที่เคยเฟื่องในเครื่อง 8 บิตเดิม ก็ถูกเครื่องที่มีหน่วยความจำมากกว่าเข้ามาสร้างแนวคิดใหม่อยู่เรื่อยๆ การใช้ภาษาในการเขียนโปรแกรมก็พัฒนาขึ้นไปเป็นภาษาโครงสร้างมากยิ่งขึ้น เช่น เทอร์โบเบสิก ปาสคาล ภาษาซี เป็นต้น

แต่สิ่งที่ยุ่งยากอย่างหนึ่งบนเครื่อง 16 บิต คือ ความซับซ้อนของฮาร์ดแวร์ในส่วนที่แสดงผลออกทางจอภาพซึ่งมีหลายแบบจนเกินไป จึงทำให้การใช้งานยุ่งยากขึ้นเมื่อต้องการเขียนภาพในงานกราฟิกไม่ว่าจะเป็นการทำบน เทอร์โบเบสิก, เทอร์โบปาสคาล 3.0 ฯลฯ ก็มีปัญหาพอ ๆ กัน (ยกเว้นท่านที่ใช้การ์ด CGA,EGA)

เทอร์โบปาสคาล 4.0 ซึ่งพัฒนาไปจากเทอร์โบปาสคาล 3.0 จนไม่สามารถนำโปรแกรมเดิมนั้นมารันบนเทอร์โบปาสคาล 4.0 ได้นั้น มีข้อดีอย่างหนึ่งคือสามารถใช้กับการ์ดแสดงผลทุกชนิดได้ การเขียน CAI ในวิชาฟิสิกส์ก็สามารถเขียนบนซอฟต์แวร์นี้ได้เช่นกัน

ก่อนอื่นเราก็ต้องศึกษาคำสั่งที่จำเป็นในการเขียนกราฟิกกันก่อน

เพื่อความสะดวกในการศึกษาคำสั่งกราฟิกดังกล่าว ผู้เขียนขออธิบายเฉพาะคำสั่งที่ใช้ในแต่ละครั้งเป็นสำคัญ คำสั่งใดที่ยังไม่ได้ใช้ก็จะไปอธิบายในบทความเรื่องต่อ ๆ ไป ดังนั้นจึงเสมือนผู้อ่านได้ศึกษากราฟิกของเทอร์โบปาสคาล 4.0 ไปพร้อม ๆ กับการได้แนวคิดของเขียนโปรแกรมไปด้วย จึงเสมือนยิงกระสุนนัดเดียวแต่ได้นกถึง 2 ตัว

เทอร์โบปาสคาล 4.0 ต้นฉบับนั้นมีทั้งหมด 3 แผ่น คือ

แผ่นที่ 1 เป็นแผ่นหลักบรรจุโปรแกรมใช้งานทั้งหมดไว้(ต้องใช้แผ่นอื่น boot ก่อน)

แผ่นที่ 2 เป็นแผ่นที่เก็บโปรแกรมตัวอย่างไว้ให้ศึกษา

แผ่นที่ 3 เป็นแผ่นที่เก็บโปรแกรมสำหรับงานด้านกราฟิกไว้ทั้งหมด มีโปรแกรมที่สำคัญคือ GRAPH.* (2 ไฟล์) คือส่วนที่ใช้งานด้านกราฟิก,*.BGI (5 ไฟล์) คือส่วนที่จะใช้ในการแสดงผลตามชนิดของการ์ดแสดงผลบนจอภาพ, *.CHR (4 ไฟล์) คือไฟล์ที่เก็บตัวอักษรไว้ 4 รูปแบบ จึงทำให้สามารถแสดงข้อความได้ถึง 5 รูปแบบ (ดูตัวอย่าง

จากรูปที่ 2)

ถ้าใช้งานด้านกราฟิกครั้งใดต้องมีแผ่นที่ 3 อยู่ในไดรฟ์ที่ใช้งานเสมอ จึงจะสามารถแสดงผลเป็นกราฟิกได้

ภาษาที่ใช้โดยส่วนใหญ่ยังคงเดิมที่เปลี่ยนแปลงไปมาก ๆ คือ คำสั่งด้านกราฟิกที่บรรจุอยู่ในไฟล์ชื่อ GRAPH.* (2 ไฟล์) นั่นเอง โปรแกรมในรูปที่ 1 เป็นโปรแกรมต้นแบบที่ผู้เขียนเขียนขึ้นมาเพื่อช่วยการใช้งานด้านกราฟิก ซึ่งบทความครั้งต่อไปจะอ้างถึงด้วย เขียนเสร็จก็ให้เก็บไว้ในชื่อ INITGR.PAS (เรียกมาใช้ในภายหลังด้วยการกด ctrl -K R และป้อนชื่อไฟล์นี้)

จากรูปที่ 1 จะมีชุดคำสั่ง (procedure) อยู่ 3 ชุด คือ Opengraph,WaitToGo,TextShow เพื่อการทำงานดังนี้

Opengraph ใช้เมื่อต้องการเข้าสู่กราฟิก (720x348 จุด บนจอโมโนโครม)

WaitToGo สำหรับรอการกดแป้น ถ้ามีการกดแป้น ESC ก็จะเป็นการจบโปรแกรม

TextShow สำหรับเปลี่ยนรูปแบบของตัวอักษร,ขนาดของตัวอักษรและการแสดงผลแนวตั้งหรือแนวนอน

มีรายละเอียดเพิ่มเติมดังนี้

คำสั่ง uses คือการเรียกใช้ไฟล์ชื่อที่ระบุหลังคำสั่งนี้คือชื่อไฟล์ที่ต้องการเรียกมาใช้งานตามรูปที่ 1 crt,graph คือชื่อไฟล์ที่ต้องการ(crt เป็นไฟล์ภายใน, graph เป็นไฟล์ภายนอกคือเป็นไฟล์ที่ต้องเรียกมาจากแผ่นดิสก์) ดูว่าคำสั่งใดอยู่ใน crt,คำสั่งใดอยู่ใน graph ได้จากหนังสือคู่มือของเทอร์โบปาสคาล 4.0 บทที่ 27 หน้า 379-503

ชุดคำสั่ง Opengraph เขียนขึ้นเมื่อให้เข้าชุดกับคำสั่ง Closegraph ซึ่งเป็นคำสั่งที่อยู่ในไฟล์ graph อยู่แล้ว เพื่อจะได้จำง่าย ๆ ว่า open เปิด, close ปิด คำสั่งในส่วนนี้ที่สำคัญคือ initgraph ซึ่งเป็นคำสั่งในไฟล์ graph ด้วยเช่นกัน แต่เป็นคำสั่งที่เรียกได้โดยการกำหนดค่าตัวแปรเท่านั้น (คือไม่สามารถเรียกได้จาก initgraph(7,0,' ');) ถ้าท่านใช้การ์ดแสดงผลเป็นชนิดอื่นก็เปลี่ยนค่าของตัวแปร grdriver,grmode ไปเป็นค่าใหม่ โดยดูจากคู่มือหน้า 433-436

ชุดคำสั่ง WaitToGo นั้นเมื่อกดแป้น ESC (ch = #27) จะเรียกคำสั่ง Halt ซึ่งเป็นการออกจากโปรแกรม

ชุดคำสั่ง TextShow ซึ่งเป็นการเรียกไฟล์ *.CHR เพื่อเปลี่ยนรูปแบบตัวอักษร (Font) ด้วยกำหนดค่า Font จาก 0-4, เปลี่ยนขนาดตัวอักษร(Size) แล้วแต่เรากำหนด และเลือกการแสดงข้อความในแนวดิ่ง ('V','v') หรือการแสดงข้อความในแนวนอน('H','h') โดยอาศัยคำสั่ง SetTextStyle ซึ่งเป็นคำสั่งในไฟล์ graph (ดูรายละเอียดได้ในหนังสือคู่มือหน้า 483-485) โดยเมื่อต้องการแสดงข้อความก็ต้องใช้คำสั่ง OutTextXY เข้ามาช่วย

เรามาดูรูปแบบของตัวอักษรที่เทอร์โบปาสคาลสร้างมาให้แล้ว โดยเขียนโปรแกรมเพิ่มเติมดังรูปที่ 2 (คือมีโปร-

```
uses    crt, Graph;

PROCEDURE Opengraph;
var
  grdriver,grmode :integer;
begin
  grdriver := 7;   grmode := 0;
  initgraph(grdriver,grmode,'');
end; { Opengraph }

PROCEDURE  WaitToGo;
var
  Ch : char;
begin
 repeat until KeyPressed;
  Ch := ReadKey;
  if Ch = #27 then
    Halt(0)
end; { WaitToGo }

PROCEDURE TextShow(Font,Size :word; Mode :char);
  begin
   case Mode of
   'V','v'  :  SetTextStyle(Font, VertDir, Size);
   'H','h'  :  SetTextStyle(Font, HorizDir, Size);
   end; { case }
  end; { TextShow }
```

รูปที่ 1
โปรแกรมต้นแบบสำหรับงานด้านกราฟิกของเทอร์โบปาสคาล 4.0

แกรมในรูปที่ 1 อยู่ด้วย) โดยใช้คำสั่ง OutTextXY มาช่วยในการกำหนดตำแหน่งที่จะให้ข้อความแสดง เช่น

textShow(1,2,'H'); Outtextxy (30,45,'TriplexFONT'); คือเลือก font ที่ 1 ขนาดเป็น 2 และแสดงผลในแนวนอนเริ่มซ้ายสุดที่จุด 30,45

textShow(4,4,'v'); outtextxy (550,25, 'GothicFont'); คือเลือก font ที่ 4 ขนาดเป็น 4 และแสดงผลในแนวตั้งมีจุดบนสุดอยู่ที่ 550,25

อยากได้อักษรแบบใดขนาดโตเท่าใดก็ใช้แนวคิดนี้เขียนโปรแกรมเอาเองได้

ตอนนี้เราก็ได้ต้นแบบใช้งานด้านกราฟิกเก็บไว้ในแผ่นดิสก์ในชื่อ initgr.pas แล้ว ซึ่งในการเขียนโปรแกรมด้านกราฟิกครั้งต่อไปเราจะเรียกไฟล์นี้เข้ามาก่อนเสมอ แล้วจึงเขียนส่วนที่เราต้องการต่อไป

เราจะลองมาเขียนกราฟิกของวิชาฟิสิกส์กันดูบ้าง

จากวารสารไมโครคอมพิวเตอร์ฉบับที่ 12 ผู้เขียนได้เสนอเรื่อง เลนส์นูนและเลนส์เว้าไปแล้ว แต่เป็นการนำเสนอในกราฟิกของแอปเปิ้ล เพื่อเป็นการศึกษาเปรียบเทียบแนวการเขียนโปรแกรมจึงขอยกมาเขียนอีกครั้ง โดยใช้เทอร์โบปาสคาล 4.0 ซึ่งโปรแกรมจะเป็นดังรูปที่ 3 แต่เพื่อท่านผู้อ่านจะได้ลองใช้ความคิดบ้างจึงทิ้งเรื่อง เลนส์เว้าให้คิดโปรแกรมเพิ่มเติมเองโดยศึกษาแนวคิดจากโปรแกรม 8 บิตในฉบับดังกล่าวเป็นแนวทาง การเขียนเรื่องเลนส์นูนนี้ก็ไม่ได้เขียนให้ทั้งหมดจึงขอให้ผู้อ่านลองพัฒนาโปรแกรมให้สามารถแสดงผลได้หลาย ๆ รูปเอาเองเช่นกัน

โปรแกรมเรื่องเลนส์นูนที่เขียนเพิ่มเติมจากรูปที่ 1 มีดังนี้

โปรแกรมในรูปที่ 3 แบ่งเป็น 2 ส่วน คือชุดคำสั่ง Lens กับส่วน main ซึ่งมีคำสั่งด้านกราฟิกอยู่หลายคำสั่ง โดยขออธิบายตามลำดับจากชุดคำสั่ง Lens ก่อน ดังนี้

ตัวแปรที่ใช้ทั้งหมดมีดังนี้

s คือระยะวัตถุ f คือความยาวโฟกัสของเลนส์นูน

point ตัวแปรเพื่อใช้ในการวาดรังสีของแสง

vy ค่าตัวแปรที่บอกถึงการเปลี่ยนแปลงของจุดในแกนตั้ง(แกน y)

so,si คือ ระยะภาพ และขนาดของภาพ(ขนาดของวัตถุในโปรแกรมกำหนดเป็น 50)

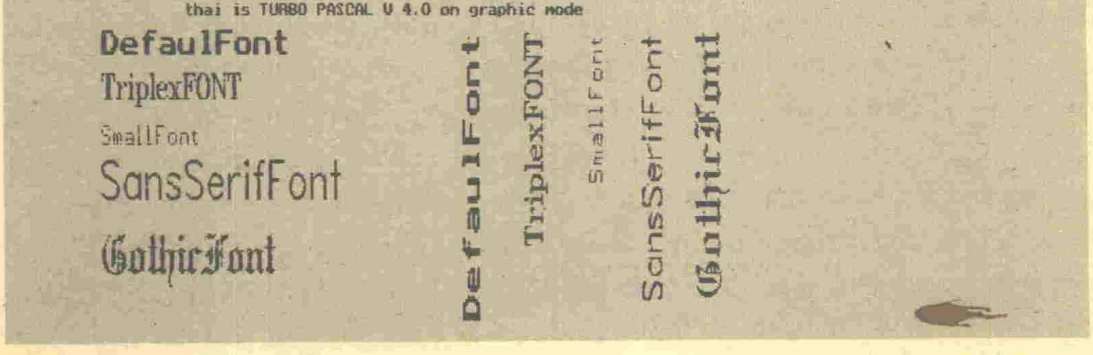
head คือตัวแปรเพื่อกำหนดการวาดรูปหัวลูกศรของภาพให้ถูกต้อง ถ้าเป็นภาพจริงค่านี้เป็น -4 และมีค่าเป็น 4 ถ้าเป็นภาพเสมือน (เมื่อค่า s<f)

คำสั่งแรกด้านกราฟิกที่ใช้คือ ellipse เช่น

ellipse(320,170,90,270,10,80); เป็นคำสั่งสำหรับวาดวงรีเต็มวงหรือส่วนหนึ่งของวงรีก็ได้ ตัวแปรที่ใช้ประกอบอธิบายได้ดังนี้

320,170 คือจุดศูนย์กลางของวงรี (320 แกน x, 170 แกน y)

```
begin
  Opengraph;
  Outtextxy(100,10,'thai is TURBO PASCAL V 4.0 on graphic mode');
  textShow(0,2,'H');   outtextxy(30,25,'DefaulFont');
  textShow(0,2,'v');   outtextxy(350,25,'DefaulFont');
  textShow(1,2,'H');   outtextxy(30,45,'TriplexFONT');
  textShow(1,2,'v');   outtextxy(400,25,'TriplexFONT');
  textShow(2,6,'H');   outtextxy(30,75,'SmallFont');
  textShow(2,6,'v');   outtextxy(450,25,'SmallFont');
  textShow(3,4,'H');   outtextxy(30,90,'SansSerifFont');
  textShow(3,3,'v');   outtextxy(500,25,'SansSerifFont');
  textShow(4,4,'H');   outtextxy(30,130,'GothicFont');
  textShow(4,4,'v');   outtextxy(550,25,'GothicFont');
  waittogo;
  CloseGraph;
end.
```



รูปที่ 2
โปรแกรมเลือกรูปแบบตัวอักษร 5 แบบ และแสดงข้อความ

```
PROCEDURE Lens(s ,f : integer);
var
 point,vy,so,si,head : integer;
begin
 head := -4;
 ellipse(320,170,90,270,10,80);              { draw lens }
 ellipse(320,170,270,90,10,80);
 line(30,170,700,170);                       { draw axes }
 textshow(3,1,'H');
 outtextxy(317,175,'O');    outtextxy(320+f,175,'f');
 outtextxy(320-f,175,'f'); outtextxy(316-2*f,175,'2f');
 moveto(320-s,170);  lineto(320-s,120);    { draw  object }
 lineto(316-s,124);  lineto(324-s,124);
 lineto(320-s,120);
 for point := 320-s to 320 do                { draw ray }
   begin  lineto(point,120);   end;
 for point := 0 to 400 do
   begin
    vy := point *50 div f; lineto(320+point,120+vy);
   end;
   moveto(320-s,120);
 for point := 0 to 650 do
   begin
    vy := point *50 div s;  lineto(320-s+point,120+vy);
   end;
 if s < f then
   begin
    moveto(320,120);  head := 4;
   for point := 0 to 300 do
    begin
     vy := point *50 div f; lineto(320-point,120-vy);
    end;
    moveto(320-s,120);
   for point := 0 to 300 do
    begin
     vy := point *50 div s; lineto(320-s-point,120-vy);
    end;
 end; { if }
  so :=(f*s) div (s-f);  si :=so *50 div s;     { draw image }
  moveto(320+so,170);    lineto(320+so,170+si);
  lineto(324+so,170+head+si); lineto(316+so,170+head+si);
  lineto(320+so,170+si);
end;  { lens }

{+++++++++++++ ( LENS : TURBO PASCAL V 4.0 ) +++++++++++++++}
begin
  Opengraph;
  textshow(1,3,'H');
  Outtextxy(150,10,'Lens  on  TURBO PASCAL V 4.0 ');
  textshow(3,2,'H');
  outtextxy(215,34,'By Arthan Sattayarak');
  lens(80,140);
  waittogo;
  CloseGraph;
end.
```

รูปที่ 3 โปรแกรมการแสดงภาพที่เกิดจากวัตถุวางไว้หน้าเลนส์นูน

90 คือมุมเริ่มต้นในการวาดวงรีนั้น ๆ (หน่วยเป็นองศา ซึ่งทำให้เรากำหนดได้ง่ายกว่าภาษาอื่น ๆ ที่มักกำหนดมุมเป็น เรเดียน)

270 คือมุมสิ้นสุดของการวาดในหน่วยองศา

10 คือระยะสูงสุดจากศูนย์กลางตามแกน x

80 คือระยะสูงสุดจากศูนย์กลางตามแกน y

นั่นคือคำสั่งนี้เป็นการวาดรูปวงรีณ จุด 320,170 โดยเริ่มวาดจากมุม 90 องศาไปสิ้นสุดที่มุม 270 องศา เป็นครึ่งวงรีในแนวตั้งซีกซ้าย(มุม 0 องศาเริ่มที่ 3 นาฬิกา โดยนับวนแบบทวนเข็มนาฬิกา)

คำสั่งถัดมาก็คือการเขียนวงรีทางซีกขวาให้ครบเป็นเลนส์นูนนั่นเอง ความจริงการวาดนี้จะใช้คำสั่งเพียงครั้งเดียวก็ได้ (ให้ลองคิดดู) ที่ผู้เขียนแยกเป็น 2 ครั้งก็เพื่อจะได้มองเห็นแนวทางการใช้คำสั่งนี้ดีขึ้นเท่านั้น

คำสั่งด้านกราฟิกถัดมาคือคำสั่ง Line ซึ่งตามโปรแกรมเป็นการลากเส้นแกนมุขสำคัญของเลนส์ คือ line(30, 170,700,170); มีรายละเอียดดังนี้

30,170 คือจุดเริ่มต้นในการลากเส้น

700,170 คือจุดสิ้นสุดของการลากเส้น

ดังนั้นตามโปรแกรมจึงเป็นการลากเส้นในแนวนอนผ่านจุดกึ่งกลางเลนส์ ต้องการลากเส้นอื่น ๆ ก็ทำได้โดยการกำหนดค่าของจุดตามที่เราต้องการ

ช่วงถัดมาก็เป็นการแสดงจุด o ,f ,f ,2f ตามตำแหน่งที่คำนวณได้เพื่อบอกระยะของจุดต่าง ๆ โดยใช้คำสั่งแบบเดียวกับรูปที่ 2

คราวนี้เราก็มาถึงส่วนการวาดรูปของวัตถุ(ช่วง draw object) มีคำสั่ง 2 คำสั่งในตอนนี้คือ Moveto และ Lineto เช่น

moveto(320, – s,170); lineto (320 – s,120);

moveto ใช้ในการเลื่อนตำแหน่งเริ่มต้นของการลงจุดกราฟิกใด ๆ ไปยังจุดใหม่ที่ระบุในวงเล็บหลังคำสั่งนี้ นั่นคือ moveto(320 – s,170); จึงเป็นการตรงไปยังจุดที่จะวาดรูปวัตถุ

lineto ใช้เมื่อต้องการลากเส้นจากจุดที่เคยลงไว้เดิมไปยังจุดที่กำหนดไว้ ท้ายคำสั่งนี้จึงได้เส้นตรงขึ้นไปข้างบนอีก 50 จุด(ซึ่งก็คือขนาดของวัตถุนั่นเอง) คำสั่ง lineto ที่เหลืออีก 3 คำสั่งก็คือการวาดรูปหัวของวัตถุ

โปรดสังเกตว่าคำสั่ง line, lineto ทำงานเกือบคล้ายกันแต่ให้การแสดงผลที่ต่างกันซึ่งในช่วงการวาดรังสีของแสง (ช่วง draw ray) จะแสดงให้เป็นประโยชน์ของคำสั่ง lineto ว่าต่างไปจากคำสั่ง line ในกรณีที่สามารถทำให้เรา

ต้นฉบับไม่มีหน้า 237

No This Page 237 In Original